Getting WITH it

Revised October 2017

Newer features of Subquery Factoring

This free training webinar is part one lesson from the Hotsos *Optimizing Oracle SQL, Intensive* (OPINT) course. Please visit our website for more information.

Presented by Ric Van Dyke
Education Director
Oracle Ace
Hotsos Enterprises, Ltd.

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Agenda

- Introduction
- Defining Subquery Factoring
- Recursive Subquery Factoring
- **`WITH`** functions

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Who is Ric?

- Oracle Ace
- Using Oracle since version 5
- Currently director of Education at Hotsos
- Prior experience:
  - Worked at Ford as a developer in Forms 2.3 and DBA
  - Worked at many companies as a consultant
  - Worked for Oracle for 10 years
    - Core DBA Senior Principal instructor
    - Education Manger, central region
    - ATS Technical Manger, north central region

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Hotsos the company…

- **R**esponse-Time **P**rofiling **M**ethod **(RPM)**
  - Focus is on the business
  - Measures exact user experience
  - Trail and error tuning obsolete
- **Products**
  - Laredo
  - HAWCS

hotsos P5

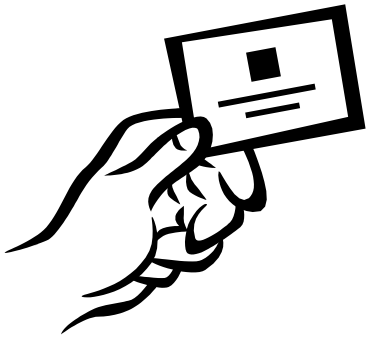
- **Education**
  - Oracle performance curriculum
  - Hotsos Symposium
- **Services**
  - On-site consulting and education
  - Remote consulting

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Contact information

- Questions?
  - Use our forum: http://support.hotsos.com/forums/index.php
    - Enter questions via the Webinars link
  - Send me email: ric.van.dyke@hotsos.com

- Inquire about classes, consulting or products:
  - Web site: www.hotsos.com
  - Phone: +1.888.8HOTSOS or +1.817.488.6200
  - Email: sales@hotsos.com



Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

`WITH` Clause, Subquery Factoring

- The **`WITH`** clause offers the benefit of reusing a query when it occurs more than once within the same SQL.
- The **`WITH`** clause allows you to give a repeated query block an alias and then reference it by that alias names multiple times.
- This avoids a re-read and re-execution of the query and can result in improvement in overall execution time and resource usage.
- Typically used when querying large volumes of data.

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

The syntax of the **WITH** clause

```
with {subquery_name_1} as (
        select statement 1
),
{subquery_name_2} as (
        select statement 2
        possibly referencing object {subquery_name_1}
)
select ...
from
        subquery_name_1  sq1,
        subquery_name_2  sq2,
        some_table       st
where   ...
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

A simple example of Using **WITH**

```
SQL> WITH
  2  revenue_per_instructor AS
  3  (SELECT instructor_id, SUM(cost) AS revenue
  4    FROM section s, course c, enrollment e
  5   WHERE s.section_id = e.section_id
  6     AND c.course_no = s.course_no
  7   GROUP BY instructor_id)
  8  SELECT *
  9    FROM revenue_per_instructor
 10   WHERE revenue > (SELECT AVG(revenue)
 11                      FROM revenue_per_instructor) ;
INSTRUCTOR_ID    REVENUE
------------- ----------
          101      51380
          103      44215
          107      35745
          108      39235
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

The Plan of the **WITH**

```
-------------------------------------------------------------------------------
| Id  | Operation                      | Name                      | Rows  |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                           |     8 |
|   1 |  TEMP TABLE TRANSFORMATION     |                           |       |
|   2 |   LOAD AS SELECT               | SYS_TEMP_0FD9D6735_1DCDD0 |       |
|   3 |    HASH GROUP BY               |                           |     8 |
|   4 |     NESTED LOOPS               |                           |   226 |
|   5 |      MERGE JOIN                |                           |    78 |
|   6 |       TABLE ACCESS BY INDEX ROWID| SECTION                   |    78 |
|   7 |        INDEX FULL SCAN         | SECT_CRSE_FK_I            |    78 |
|*  8 |       SORT JOIN                |                           |    30 |
|   9 |        TABLE ACCESS FULL       | COURSE                    |    30 |
|* 10 |      INDEX RANGE SCAN          | ENR_SECT_FK_I             |     3 |
|* 11 |   VIEW                         |                           |     8 |
|  12 |    TABLE ACCESS FULL           | SYS_TEMP_0FD9D6735_1DCDD0 |     8 |
|  13 |    SORT AGGREGATE              |                           |     1 |
|  14 |     VIEW                       |                           |     8 |
|  15 |      TABLE ACCESS FULL         | SYS_TEMP_0FD9D6735_1DCDD0 |     8 |
-------------------------------------------------------------------------------
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Without the **WITH** clause

```
SQL> SELECT instructor_id, SUM(cost) AS revenue
  2    FROM section s, course c, enrollment e
  3   WHERE s.section_id = e.section_id
  4     AND c.course_no = s.course_no
  5   GROUP BY instructor_id
  6  HAVING SUM(cost) > (SELECT AVG(SUM(cost))
  7    FROM section s, course c, enrollment e
  8   WHERE s.section_id = e.section_id
  9     AND c.course_no = s.course_no
 10   GROUP BY instructor_id) ;
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

```
-------------------------------------------------------------------------
| Id  | Operation                          | Name            | Rows  |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                 |     1 |
|*  1 |  FILTER                            |                 |       |
|   2 |   HASH GROUP BY                    |                 |     1 |
|   3 |    NESTED LOOPS                    |                 |   226 |
|   4 |     MERGE JOIN                     |                 |    78 |
|   5 |      TABLE ACCESS BY INDEX ROWID   | SECTION         |    78 |
|   6 |       INDEX FULL SCAN              | SECT_CRSE_FK_I  |    78 |
|*  7 |      SORT JOIN                     |                 |    30 |
|   8 |       TABLE ACCESS FULL            | COURSE          |    30 |
|*  9 |     INDEX RANGE SCAN               | ENR_SECT_FK_I   |     3 |
|  10 |   SORT AGGREGATE                   |                 |     1 |
|  11 |    SORT GROUP BY                   |                 |     1 |
|  12 |     NESTED LOOPS                   |                 |   226 |
|  13 |      MERGE JOIN                    |                 |    78 |
|  14 |       TABLE ACCESS BY INDEX ROWID| SECTION         |    78 |
|  15 |        INDEX FULL SCAN             | SECT_CRSE_FK_I  |    78 |
|* 16 |       SORT JOIN                    |                 |    30 |
|  17 |        TABLE ACCESS FULL           | COURSE          |    30 |
|* 18 |      INDEX RANGE SCAN              | ENR_SECT_FK_I   |     3 |
-------------------------------------------------------------------------
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Comparing the two queries

TYPE	NAME	withex:nonewith	withex:with	DIFFERENCE
Stats	buffer is pinned count	308	157	151
	consistent gets	778	21	757
	db block changes	0	11	-11
	db block gets	0	10	-10
	execute count	6	6	0
	redo size	0	948	-948
	session logical reads	778	31	747
	sorts (memory)	3	1	2
	sorts (rows)	286	30	256
	table fetch by rowid	156	78	78
	table scan blocks gotten	10	8	2
	table scans (short tables)	2	3	-1

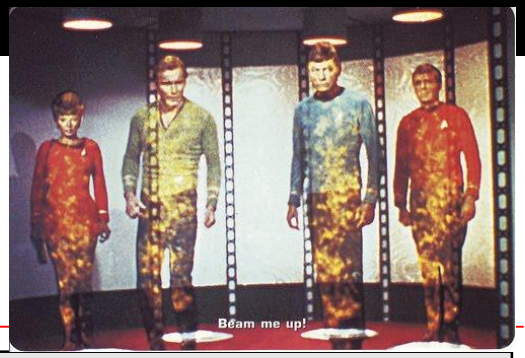
Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Referencing Preceding Subqueries

```
WITH regional_sales AS
(
SELECT region, NVL(SUM(order_amt),0) AS total_sales
  FROM sales
 GROUP BY region
),
top_regions AS
(
SELECT region
  FROM   regional_sales
 WHERE  total_sales > (SELECT SUM(total_sales)/3 AS one_third_sales
                       FROM   regional_sales)
)
SELECT region, product, COUNT(*) AS product_units,
       SUM(order_amt) AS product_sales
  FROM sales
 WHERE region IN (SELECT region FROM top_regions)
 GROUP BY region, product
/
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

The **MATERIALIZE** hint




```
WITH inline_view AS
(
SELECT /*+ materialize */ region, SUM(order_amt) AS total_orders
  FROM sales
 GROUP BY region
)
SELECT *
  FROM inline_view
/
```

```
-------------------------------------------------------------------------------------------
|Id|Operation                 |Name                      |Rows |Bytes |Cost (%CPU)|Time    |
-------------------------------------------------------------------------------------------
| 0|SELECT STATEMENT          |                          |  30 |  900 |  683  (13)|00:00:09|
| 1| TEMP TABLE TRANSFORMATION|                          |     |      |           |        |
| 2|  LOAD AS SELECT          |SYS_TEMP_0FD9D67B7_4F95D7|     |      |           |        |
| 3|   HASH GROUP BY          |                          |  30 |  330 |  681  (13)|00:00:09|
| 4|    TABLE ACCESS FULL     |SALES                     |1000K|   10M|  612   (3)|00:00:08|
| 5|  VIEW                    |                          |  30 |  900 |    2   (0)|00:00:01|
| 6|   TABLE ACCESS FULL      |SYS_TEMP_0FD9D67B7_4F95D7|  30 |  330 |    2   (0)|00:00:01|
-------------------------------------------------------------------------------------------
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Recursive Subquery Factoring (11.2)


```
WITH
  reports_to_102 (eid, emp_last, mgr_id, reportLevel) AS
  (
     SELECT employee_id, last_name, manager_id,
            1 reportLevel
     FROM employees
     WHERE employee_id = 102
   UNION ALL
     SELECT e.employee_id, e.last_name, e.manager_id,
            reportLevel+1
     FROM reports_to_102 r, employees e
     WHERE r.eid = e.manager_id
  )
SELECT eid, emp_last, mgr_id, reportLevel
FROM reports_to_102
ORDER BY reportlevel, eid
/
```

ANCHOR member

RECURSIVE member

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Recursive Subquery Factoring – Plan

```
Plan hash value: 3519480695

-------------------------------------------------------------------------
| Id  | Operation                                    | Name           |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                             |                |
|   1 |  SORT ORDER BY                               |                |
|   2 |   VIEW                                       |                |
|   3 |    UNION ALL (RECURSIVE WITH) BREADTH FIRST|                |
|   4 |     TABLE ACCESS BY INDEX ROWID              | EMPLOYEES      |
|*  5 |      INDEX UNIQUE SCAN                       | EMP_EMP_ID_PK  |
|*  6 |     HASH JOIN                                |                |
|   7 |      RECURSIVE WITH PUMP                     |                |
|   8 |      TABLE ACCESS FULL                       | EMPLOYEES      |
-------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - access("EMPLOYEE_ID"=102)
   6 - access("R"."EID"="E"."MANAGER_ID")
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Using **START WITH, CONNECT BY PRIOR** syntax

```
SELECT employee_id, last_name, manager_id, LEVEL
      FROM employees
      START WITH employee_id = 102
      CONNECT BY PRIOR employee_id = manager_id
      ORDER by level, employee_id
/
```

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Using Connect by Prior – Plan

```
-----------------------------------------------------
| Id  | Operation                      | Name         |
-----------------------------------------------------
|   0 | SELECT STATEMENT               |              |
|   1 |  SORT ORDER BY                 |              |
|*  2 |   CONNECT BY WITH FILTERING    |              |
|   3 |    TABLE ACCESS BY INDEX ROWID| EMPLOYEES     |
|*  4 |     INDEX UNIQUE SCAN          | EMP_EMP_ID_PK |
|*  5 |    HASH JOIN                   |              |
|   6 |     CONNECT BY PUMP            |              |
|   7 |     TABLE ACCESS FULL          | EMPLOYEES     |
-----------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("MANAGER_ID"=PRIOR NULL)
   4 - access("EMPLOYEE_ID"=102)
   5 - access("connect$_by$_pump$_002"."PRIOR employee_id
"="MANAGER_ID")
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Difference

TYPE	NAME	with:recur	with:nonrecur	DIFFERENCE
Latch	cache buffers chains	4,057	4,079	-22
	row cache objects	97	94	3
	shared pool	27	64	-37
Stats	buffer is pinned count	0	3	-3
	consistent gets	1,499	1,499	0
	consistent gets direct	0	0	0
	db block changes	120	0	120
	db block gets	241	0	241
. . .				
	sorts (disk)	0	0	0
	sorts (memory)	6	7	-1
	sorts (rows)	7,682	11,523	-3,841
	table fetch by rowid	1	1	0
	table scan blocks gotten	1,480	1,480	0
	table scans (long tables)	0	0	0
	table scans (short tables)	4	4	0
Time	elapsed time (centiseconds)	37	40	-3

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Another example:
Getting the execution plan from **v$sql_plan**

```
column query_plan format a50
WITH
get_plan (query_plan, id, planlevel) as
( select operation||' '||options||' '||object_name query_plan,
         id, 0 planlevel
  from v$sql_plan where id = 0 and sql_id = 'g76m74pycbct4' and
child_number = 0
union all
  select lpad(' ',2*planlevel+2)||p.operation||' '||
         p.options||' '||p.object_name query_plan, p.id, planlevel+1
  from get_plan g, v$sql_plan p where g.id = p.parent_id and
       sql_id = 'g76m74pycbct4' and child_number = 0)
SELECT ID, QUERY_PLAN FROM GET_PLAN ORDER BY id
/
```

```
                ID QUERY_PLAN
------------------ --------------------------------------------------
                 0 SELECT STATEMENT
                 1   TABLE ACCESS BY INDEX ROWID BATCHED EMPLOYEES
                 2     INDEX RANGE SCAN EMPL_FNAME_IDX
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

DEPTH first or **BREADTH** first?

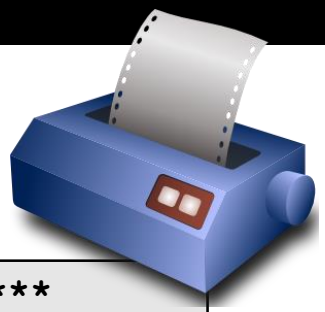
```
WITH
  org_chart (eid, emp_last, mgr_id, reportLevel) AS
  (
     SELECT empno, ename, mgr,1 reportLevel
     FROM emp
     WHERE job='PRESIDENT'
   UNION ALL
     SELECT e.empno, e.ename, e.mgr,
            reportLevel+1
     FROM org_chart r, emp e
     WHERE r.eid = e.mgr
  )
  SEARCH DEPTH FIRST BY eid SET order1
SELECT lpad(' ',2*reportLevel)||eid emp_no, emp_last
FROM org_chart
ORDER BY order1
/
```

```
SEARCH BREADTH FIRST BY eid SET order1
```

rsf1.sql

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Example output



```
**********************
********* DEPTH FIRST
**********************

EMP_NO                EMP_LAST
--------------------  --------
  7839                KING
    7566              JONES
      7788            SCOTT
        7876          ADAMS
      7902            FORD
        7369          SMITH
    7698              BLAKE
      7499            ALLEN
      7521            WARD
      7654            MARTIN
      7844            TURNER
      7900            JAMES
    7782              CLARK
      7934            MILLER
```

```
************************
********* BREADTH FIRST
************************

EMP_NO                EMP_LAST
--------------------  --------
  7839                KING
    7566              JONES
    7698              BLAKE
    7782              CLARK
      7499            ALLEN
      7521            WARD
      7654            MARTIN
      7788            SCOTT
      7844            TURNER
      7900            JAMES
      7902            FORD
      7934            MILLER
        7369          SMITH
        7876          ADAMS
```

BREADTH appears to be the default

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

`WITH` functions (12c)

- Starting 12c you can define a function in a `WITH` clause
  - Function scope is within the statement
  - This can run much faster then calling the same function
- You can define procedures as well
  - Not likely to be used much
  - Can't be directly called from SQL

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Not using a with function

```
SQL> column line format 9999
SQL> column text format a80
SQL> select line, text from user_source where name = 'RTN_DATE_CONVERT'
  2  order by line;

 LINE TEXT
----- --------------------------------------------------------------------------
    1 function rtn_date_convert (p_unix_gmt in number)
    2 return date deterministic
    3 as
    4   v_date  date;
    5 begin
    6
    7    v_date := to_date('01/01/1970','mm/dd/yyyy') + (p_unix_gmt/86400);
    8
    9    RETURN v_date;
   10 end ;

SQL> get with_nofun
  1  select count(*) from ord2
  2* where rtn_date_convert(gmt_order_date) > sysdate - 7000
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Using a with function

```
SQL> get with_fun
  1  with function rtn_date_convert2 (p_unix_gmt in number)
  2    return date
  3    as
  4     v_date  date;
  5    begin
  6       v_date := to_date('01/01/1970','mm/dd/yyyy') + (p_unix_gmt/86400);
  7       RETURN v_date;
  8    end ;
  9  select count(*) from ord2
 10* where rtn_date_convert2(gmt_order_date) > sysdate - 7000
SQL>
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Stat Lines from 10046 trace (each were run several times)

- **WITH_NOFUN**:

```
STAT #387078408 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT
AGGREGATE (cr=191 pr=0 pw=0 time=84395 us)'

STAT #387078408 id=2 cnt=12890 pid=1 pos=1 obj=124647
op='TABLE ACCESS FULL ORD2 (cr=191 pr=0 pw=0 time=82185 us
cost=60 size=270690 card=12890)'
```

- **WITH_FUN**:

```
STAT #381599352 id=1 cnt=1 pid=0 pos=1 obj=0 op='SORT
AGGREGATE (cr=191 pr=0 pw=0 time=45235 us)'

STAT #381599352 id=2 cnt=12890 pid=1 pos=1 obj=124647
op='TABLE ACCESS FULL ORD2 (cr=191 pr=0 pw=0 time=43000 us
cost=60 size=270690 card=12890)'
```


Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

Restrictions on Subquery Factoring

- First available in v9.2.
- Can't be nested.
- In a query with set operators, the set operator subquery cannot contain the **`WITH`** clause.
- Prior to 11g, if you define one you must use it.
- Can be in view, but parentheses can be tricky – No outer parentheses.
- Can not always be materialized - if you use any object columns in your subqueries, Oracle will not materialize the dataset, even if you hint it.

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.

hotsos℠

Oracle. Performance. Now.

Thanks for attending!

Contact us!

+1.888.8HOTSOS

+1.817.488.6200

Send us an email at:

sales@hotsos.com

Copyright © 1999–2017 by Hotsos Enterprises, Ltd.